



LAB 3-2

SOCKET PROGRAMMING 2

File Transfer

LAB 3-2

- Write file transfer client and server in language C.
- Please modify the sample code to achieve the goals.
- Follow the hints of “TODO”
- All the knowledge you need is in the Lab3 slides.
(For more information, please read online [library document](#))

File Directories



Requirement - Server

- Server

1. User can assign **port** to server
2. Returns files list to client when the client connects immediately to the server
3. Returns files which client wants to download.

Requirement - Client

■ Client

1. User can assign **IP address** and **port** to connect the server with command line argument.
2. Support user inputs the file name to download file from server, **one file per input**.
3. Input **".exit"** to disconnect from server.

Commands

1. make

= gcc server.c -o server
+ gcc client.c -o client

2. ./server 8888

PORT NUM

3. ./client 127.0.0.1 8888

IP

PORT

Well-known Ports	0 – 1023
Registered ports	1024 - 49151
Dynamic ports	49152 - 65535



SERVER SETUP

Set Up Socket Connection. TODO 1

```
/**
 * TODO 1:
 * preparing sockaddr_in
 */
bzero(&svr_addr, sizeof(svr_addr));
svr_addr.sin_family = /* Protocol stack */;
svr_addr.sin_addr.s_addr = htonl(INADDR_ANY);
svr_addr.sin_port = /* Bind port */;

****/
```


Set Up Socket Connection. TODO 1 (Hint)

- Make use of input of `int main(int argc, char *argv[])`
- <https://blog.gtwang.org/programming/c-cpp-tutorial-argc-argv-read-command-line-arguments/>
- `atoi()`: char 轉為 int
- http://tw.gitbook.net/c_standard_library/c_function_atoi.html

Set Up Socket Connection. TODO 2 - 3

```
/**
 * TODO 2:
 * bind the socket to port, with prepared sockaddr_in structure
 */
```

```
/***/
```

```
/**
 * TODO 3:
 * listen on socket
 */
```

```
/***/
```

Set Up Socket Connection. TODO 4

```
while(1) {  
    /**  
     * TODO 4:  
     * accept client connections  
     */  
  
    /***/  
  
    printf("[INFO] Connection accepted (id: %d)\n", cli_fd);  
    printf("[INFO] Client is from %s:%d\n", inet_ntoa(cli_addr.sin_addr), ntohs(cli_addr.sin_port));  
  
    connection_handler(cli_fd);  
  
    close(cli_fd);  
}
```

Get filenames in remote_storage directory, and write those to client. TODO 5

```
void file_listing_handler(int sockfd) {
    DIR* pDir;                // directory
    struct dirent* pDirent = NULL; // directory and children file in this dir
    char buf[MAX_SIZE];        // buffer to store msg

    printf("[INFO] List file to client\n");

    /* open remote storage directory */
    if ((pDir = opendir("./remote_storage")) == NULL) {
        perror("Open directory failed\n");
    }

    /* traversing files in remote storage and sending filenames to client */
    memset(buf, '\0', MAX_SIZE);
    while ((pDirent = readdir(pDir)) != NULL) {
        /* ignore current directory and parent directory */
        if (strcmp(pDirent->d_name, ".") == 0 || strcmp(pDirent->d_name, "..") == 0) {
            continue;
        }

        /**
         * TODO 5:
         * send filenames to client
         * //server client 間如何達成協議，彼此知道要write/read幾次為關鍵！
         */
    }

    closedir(pDir);
    /***/
}
```

TODO 5 (Hint)

- opendir(dir_name) return pointer of DIR
- readdir(pDIR) return pointer of struct of dirent
- dirent.h library document

```
struct dirent {  
    ino_t      d_ino;      /* inode number */  
    off_t      d_off;      /* offset to the next dirent */  
    unsigned short d_reclen; /* length of this record */  
    unsigned char d_type;   /* type of file; not supported  
                           by all file system types */  
    char        d_name[256]; /* filename */  
};
```

Read the filename which client want to download. (continue until receive ".exit")

```
/* read request filename from client*/
while ((read(sockfd, filename, MAX_SIZE)) > 0) {
    /* client want to exit*/
    if (strcmp(filename, ".exit") == 0) {
        break;
    }
    printf("[INFO] Client send `%s` request\n", filename);

    /* sending this file */
    file_sending_handler(sockfd, filename);
    memset(filename, '\0', MAX_SIZE);
}
```

Tell the size of the file to client after getting the filename. TODO 6

```
sprintf(path, "remote_storage/%s", filename);
fp = fopen(path, "rb");
if (fp) {
    /* send start downloading message */
    memset(buf, '\0', MAX_SIZE);
    sprintf(buf, "[ - ] Downloading `%s` ... \n", filename);
    if (write(sockfd, buf, MAX_SIZE) < 0) {
        printf("Send downloading message failed");
        return;
    }

    /* get file size, store in file_size */
    fseek(fp, 0, SEEK_END);
    file_size = ftell(fp);
    rewind(fp);

    memset(buf, '\0', MAX_SIZE);
    sprintf(buf, "%d", file_size);

    /**
     * TODO 6:
     * send file size to client
     */
}
```

Write file segment to client by using buf.

TODO 7

```
/* read file data and send to client */
write_sum = 0;
while (write_sum < file_size) {

    /* read local file to buf */
    memset(buf, '\0', MAX_SIZE);
    write_byte = fread(&buf, sizeof(char), MAX_SIZE, fp);

    /**
     * TODO 7:
     * send file data to client
     */

    /***/

    write_sum += write_byte;
}

fclose(fp);
```


TODO 7 (Hint.)

- Must let server and client know the time of read/write.
- Fread library document



CLIENT SETUP

Set Up Socket Connection. TODO 1

```
/**
 * TODO 1:
 * preparing sockaddr_in
 */
bzero(&svr_addr, sizeof(svr_addr));
svr_addr.sin_family = /* protocol stack */;
svr_addr.sin_port = /* bind port */;
if (inet_pton(AF_INET, argv[1], &svr_addr.sin_addr) <= 0) {
    perror("Address converting fail with wrong address argument");
    return 0;
}

/***/
```

Read/Write correspond on Server. TODO 2 - 3

Skip

Receive the file segment from server. TODO 4

```
read_sum = 0;
fp = fopen(path, "wb");
if (fp) {
    while (read_sum < file_size) {
        memset(buf, '\0', MAX_SIZE);

        /**
         * TODO 4:
         * receive file data from server
         */

        /***/

        /* write file to local disk*/
        fwrite(&buf, sizeof(char), read_byte, fp);
        read_sum += read_byte;
    }
    fclose(fp);

    /* receive download complete message */
    memset(buf, '\0', MAX_SIZE);
    read(sockfd, buf, MAX_SIZE);
    printf("%s", buf);
}
```

Deadline

- Please also upload your **code** and a **screenshot** of your result to iLMS system.
- Deadline: before **2019/11/10 23:59 (Sun.)**
- One week delay, taking **20%** off.
- Two weeks delay, taking **40%** off
- After **2019/11/25**, your submission is NOT accepted.

Expected Result

```
1. vicky@HsnlMacbookAir: ~/Repo/socket-programming/file-transfer (zsh)

X .server (server) X .file-transfer (zsh)

vicky@HsnlMacbookAir ~/Repo/socket-programming/file-transfer <master>
$ ./server 8888
File transfer server started
Maximum connections set to 5
Listening on 0.0.0.0:8888
Waiting for client...

[INFO] Connection accepted (id: 4)
[INFO] Client is from 127.0.0.1:56453
[INFO] Send hello msg to client
[INFO] List file to client
[INFO] Client send `test.txt` request
[INFO] Client send `test.png` request
[INFO] Client send `lab3_2_spec.pdf` request
[INFO] Connection closed (id: 4)

vicky@HsnlMacbookAir ~/Repo/socket-programming/file-transfer <master>
$ ./client 127.0.0.1 8888
[✓] Connect to server.
[✓] Server reply!
-----
Files on server:
lab3_2_spec.pdf
test.png
test.txt
-----
Enter the filename: test.txt
[-] Downloading `test.txt` ...
[✓] Download successfully!
-----
Enter the filename: test.png
[-] Downloading `test.png` ...
[✓] Download successfully!
-----
Enter the filename: lab3_2_spec.pdf
[-] Downloading `lab3_2_spec.pdf` ...
[✓] Download successfully!
-----
Enter the filename: .exit
[x] Socket closed
vicky@HsnlMacbookAir ~/Repo/socket-programming/file-transfer <master*>
$
```